Android en action

Abdelkader Lahmadi

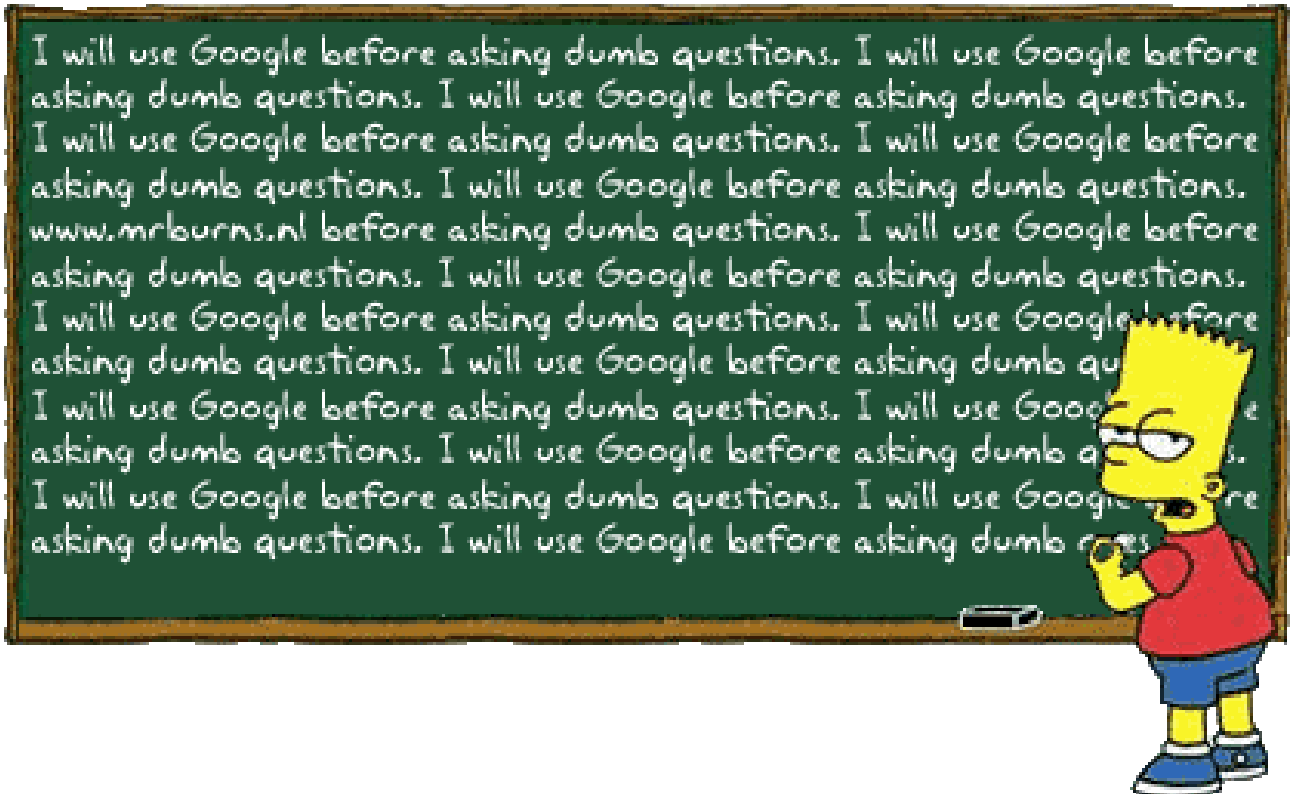
Université de Lorraine - 2013

Plan

- La boîte à outils
  - JDK
  - SDK Android
  - Eclipse
  - Plugin Android pour Eclipse
- Architecture et composants
- La main à la pâte
  - Hello Android
  - La blague du jour
- Pour aller plus loin
  - Quelques démos : Hinky, Occasus, Flowoid
  - Accès aux services sensoriels : localisation, accéléromètre

Références

- La mine d'or : http://developer.android.com/guide/index.html
- Astuces, exemples, tutoriaux: http://www.anddev.org/index.php



Installation : JDK

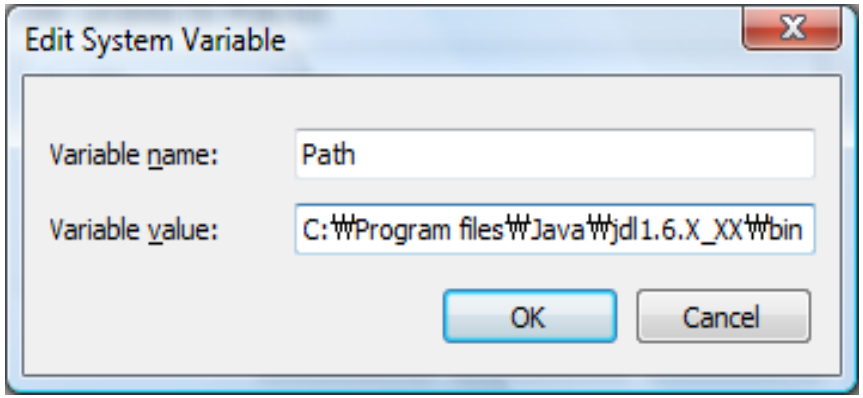
"What Java Do I Need?" You must have a copy of the **JRE** (Java Runtime Environment) on your system to **run** Java applications and applets. To **develop** Java applications and applets, you need the **JDK** (Java Development Kit), which includes the JRE.

- JDK (Java Developement Kit)
  - Indispensable pour Eclipse et pour développer des applications Android
- Etapes d'installation
  - Télécharger le JDK (version 6) : http://www.oracle.com/technetwork/java/index.html
  - Installer le paquetage téléchargé
  - Mettre à jour le chemin de recherche des exécutables
    - Pour Linux

```
# vi ~/.bash_profile
PATH=$PATH:/usr/java/jdk1.6.X_XX/bin
# source ~/.bash_profile
# java
Usage: java [-options] class [args...] (to execute a class)
   or  java [-options] -jar jarfile [args...] (to execute a jar file)
```

Installation JDK

- Pour windows
  - Modifier les variables d'environnement



- Test

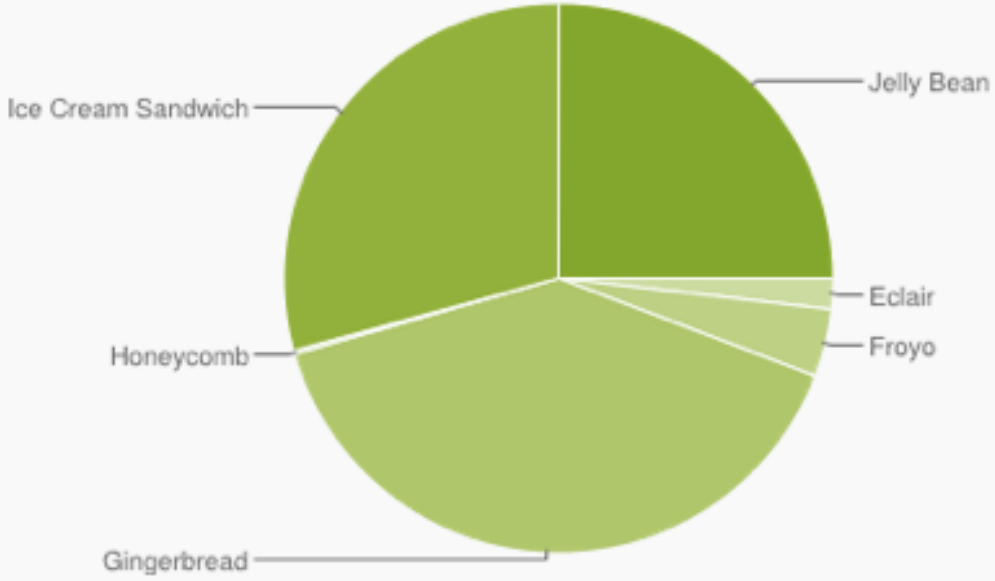
```
# javac  -version
javac 1.6.0_15
```

Le SDK Android

- Contenu
  - Outils de développement : adb, émulateur, graphiques (layoutopt, Draw 9-patch), aapt (packages), aidl (IPC), sqlite3 (database), dx (android bytecodee), avd (virtual device), etc
  - Images systèmes : 2.1, 2.2, 2.3, 3.0,..,4.2
  - Exemples de codes et d'applications
  - Documentation : API

Les SDK Android : répartition des versions

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	4.0%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	39.7%
3.2	Honeycomb	13	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.3%
4.1.x	Jelly Bean	16	23.0%
4.2.x		17	2.0%

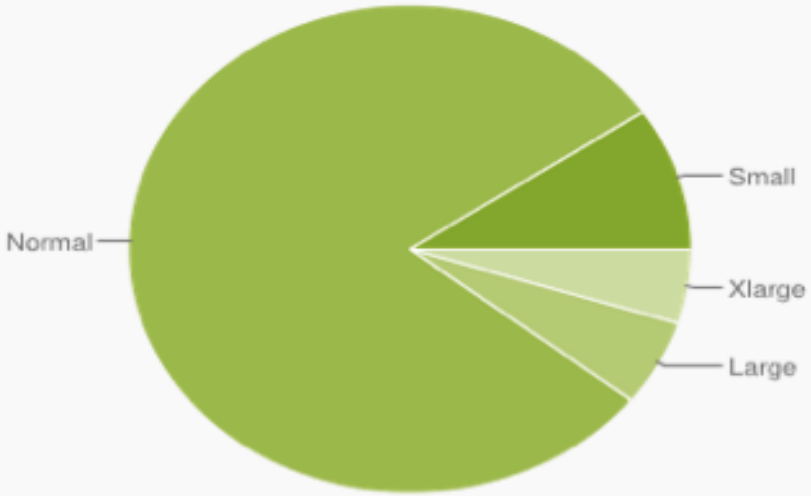


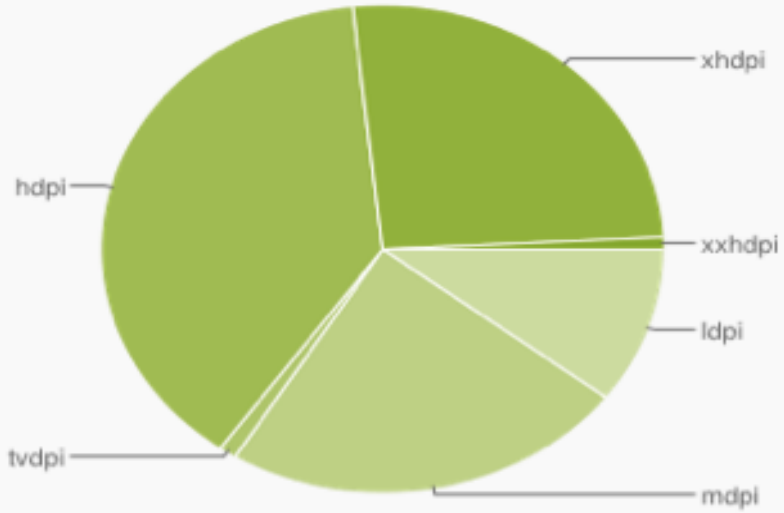
Data collected during a 14-day period ending on April 2, 2013.
Any versions with less than 0.1% distribution are not shown.

Taille et densité des écrans : répartition

- Taille de l'écran : mesure de la diagonale de l'écran physique
- Densité de l'écran : nombre de pixels par pouce (dpi : dots per inch)

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	9.5%						9.5%
Normal	0.1%	16.1%		37.9%	25.0%	0.8%	79.9%
Large	0.7%	2.7%	1.0%	0.5%	0.8%		5.7%
Xlarge	0.1%	4.6%		0.1%	0.1%		4.9%
Total	10.4%	23.4%	1.0%	38.5%	25.9%	0.8%	





Data collected during a 14-day period ending on April 2, 2013
Any screen configurations with less than 0.1% distribution are not shown.

Installation : SDK Android

- Télécharger le SDK : http://developer.android.com/sdk/
- Disponible pour : Linux, Windows, Mac OS X (intel)
- Décompresser le fichier dans un répertoire
- Mettre à jour la variable d'environnement PATH avec le chemin <SDK ROOT>/tools : .bash_profile (Linux) ou Variables d'environnements (Windows)
- Test

```
# dx  --version
dx version 1.1
```

Aperçu du SDK Android

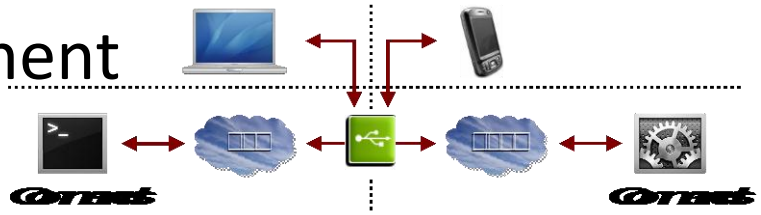
- <sdk/tools> : outils de test de deboguage (emulator, ddms, proguard, sqlite3)
- <sdk/platform-tools> : outils de développement (adb, aidl, appt)
- <sdk/docs> : documentation de l'API android
- <sdk/platforms> : sdk par plateforme Android (android.jar, samples, image système (x86,ARM))

Le SDK android

- Des outils de développement
  - Emulateur : basé sur QEMU
    - Équipement mobile virtuel de type ARM
    - Exécution des applications Android
  - ADT : plugin Eclipse (IDE)
    - Accès aux outils de dev depuis Eclipse
    - Création des projets Android
    - Compilation et exécution des applications
    - Installation des paquetages APK sur l'équipement ou l'émulateur

Encore des outils de dev

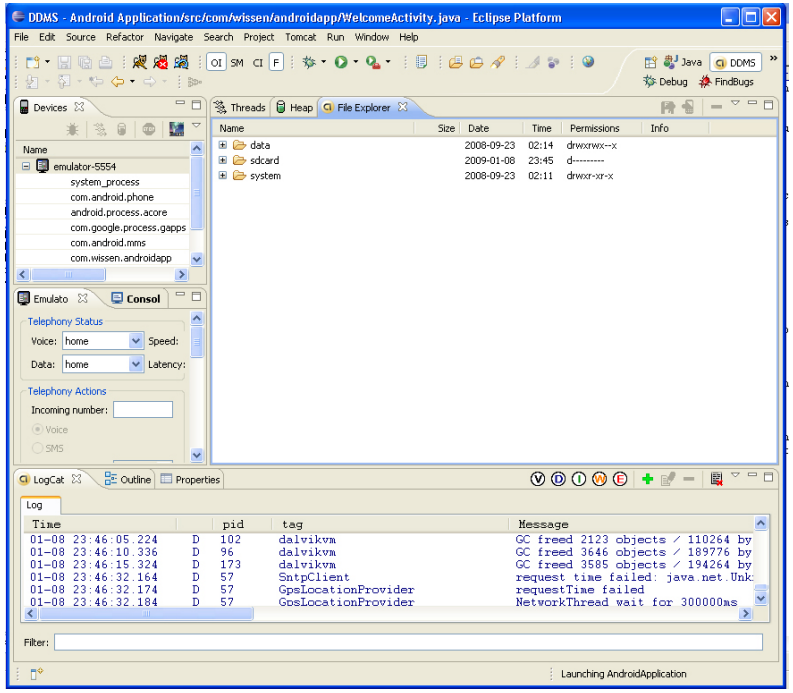
- Android
  - Gérer l'installation des composants du SDK, les équipements virtuels (AVD), lancer les émulateurs, création des projets, …
- Android Debug Bridge (adb)
  - Installation des packages des applications sur émulateur ou équipement
  - Accès shell à un équipement
  - Copie des fichiers



Encore des outils de dev

- Dalvik Debug Monitor Service (ddms)
  - Gestion de processus dans un émulateur ou un équipement mobile, aide au débogage

- Fonctions
  - Tuer les processus
  - Sélection d'un processus à déboguer
  - Visualisation de l'état du système : heap, threads
  - Prise de screenshot de l'émulateur ou l'équipement
  - Envoyer des appels ou des SMS vers l'émulateur



Encore des outils de dev

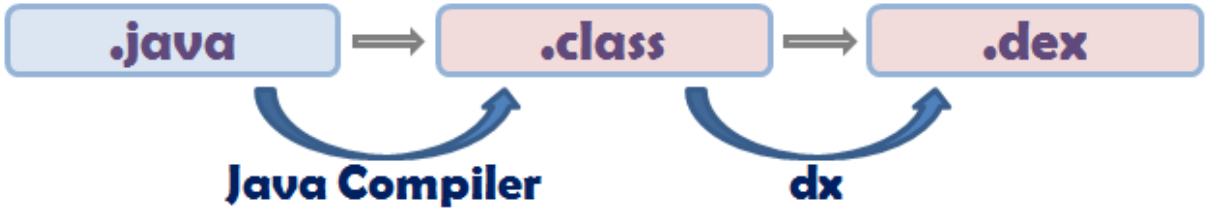
- Dmtracedump
  - Affichage graphique des traces d'exécution d'une application : Classe Debug, DDMS
- Draw 9-patch
  - Éditeur graphique des images de type NinePatch
  - NinePatch : définir des images ajustable automatiquement en fonction de la taille de l'écran
- Android Asset Packaging tool (aapt)
  - Création des paquetages d'installation des applications : binaires et ressources

Encore des outils de dev

- Android Interface Description Language (aidl)
  - langage de description des interfaces de communication inter-processus
  - Utilisé par les services Android
- sqlite3
  - Accès aux bases de données de type SQLite
- Traceview
  - visualisation graphique des traces de log d'une application Android

Encore des outils de dev

- mksdcard
  - Création des images disque utilisés par l'émulateur : émuler la présence d'une carte de stockage (SD card par exemple)
- dx : transformer le bytecode Java en bytecode android (.dex)



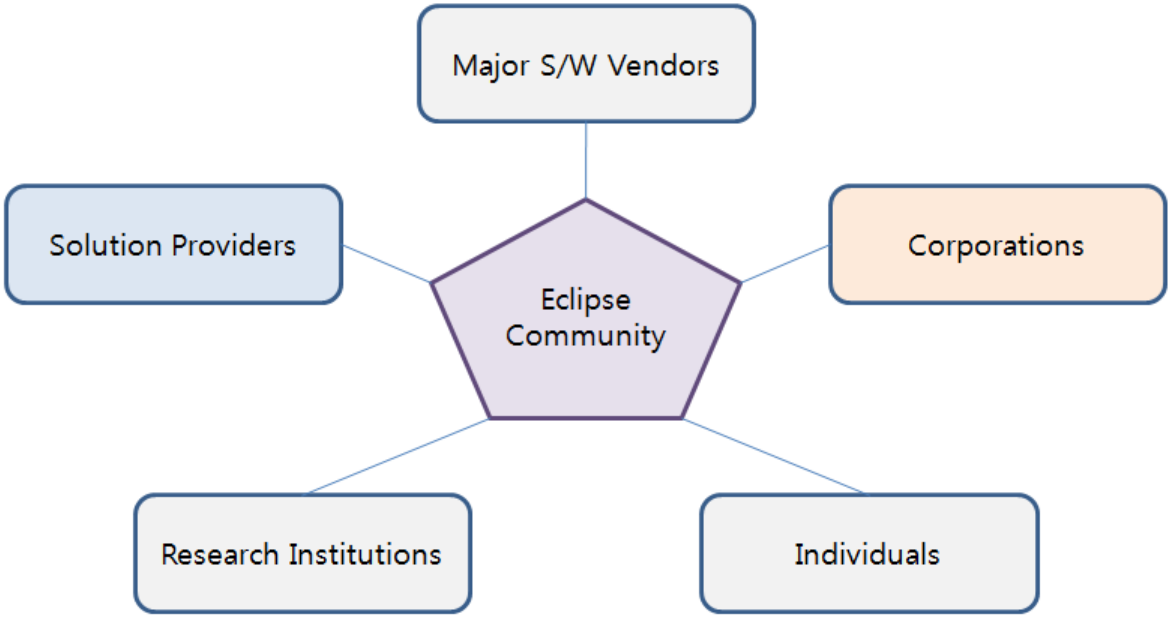
Encore des outils de dev

- UI/Application Excerciser Monkey
  - Tester la robustesse de l'application : génération des événements systèmes ou/et utilisateurs aléatoires (cliques souris, touches, gestes)
  - Stresser l'application sous test
- Proguard
  - Optimisation de code, obfuscation
  - Rendre plus difficile la rétro-ingénierie de votre application

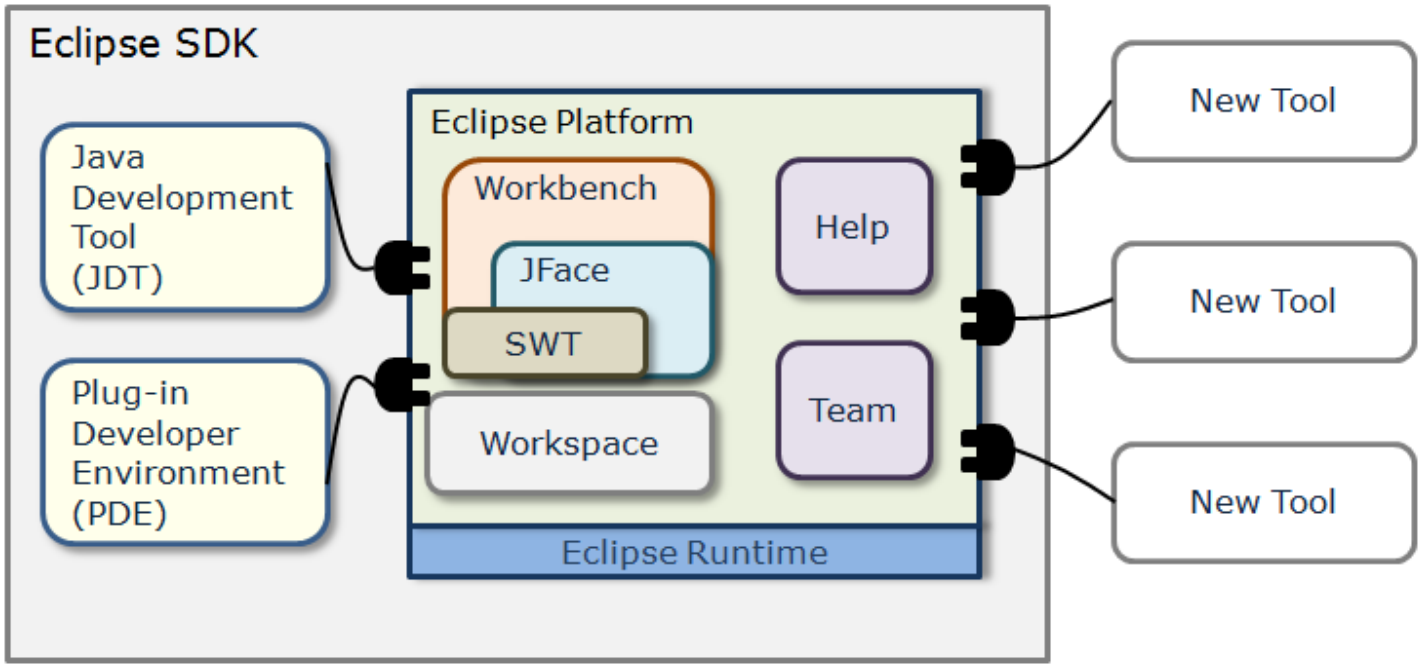
Pour plus de détails, vous pouvez consulter :
http://developer.android.com/guide/developing/tools/index.html

Installation d'Eclipse

- Eclipse : www.eclipse.org
  - Integrated Development Toolkit (IDE)
  - Projet Open Source : communauté très active



La plate-forme Eclipse



- Architecture ouverte pour intégrer de nouveaux greffons (plugins) : enrichir les fonctions d'Eclipse

Installation d'Eclipse

- Télécharger Eclipse *http://eclipse.org/mobile/*Eclipse IDE for Mobile Developers (version Juno)
- Décompresser l'archive : par example dans */home/lahmadi/eclipse*
- *Mettre à jour la variable PATH : .bash_profile (Linux)*
- *Test*
  - lancer Eclipse

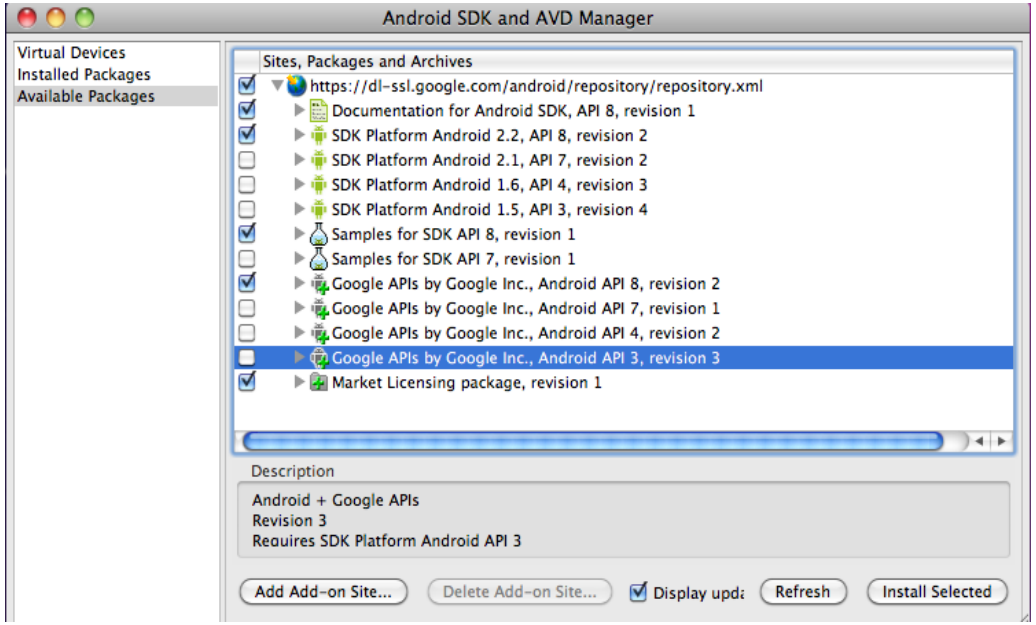
Installation de ADT

- Android Development Tools (ADT) : plugin android pour Eclipse
- Etapes d'installation depuis Eclipse
- Lancer Eclipse, ensuite cliquer sur Help>Install New Software
  - Cliquer sur le bouton add… et ajouter les adresses suivantes :
    - https://dl-ssl.google.com/android/eclipse
    - Ou (http://dl-ssl.google.com/android/eclipse/)
  - Sélectionner Developer Tools : Android DDMS, Android Development Tools
  - Cliquer sur le bouton Next, Next, Finish
  - Relancer Eclipse
  - Modifier les préférences d'Eclipse pour prendre en compte le SDK Android
    - Choisissez Fenêtre>Préférences …
    - Sélectionnez Android
    - SDK Location : indiquer le répertoire d'installation du SDK Android
    - Cliquer Apply et OK

Android SDK and AVD Manager

- Il faut aussi installer une plate-forme cible d'Android : 2.2, 2.3, 3.0, …, 4.2
- Un outil est fourni par ADT

1. Window> Android SDK and AVD Manager
2. Available Packages
   - Choisissez votre plate-forme Android cible et d'autres composants à installer
   - Cliquer Install Selected
3. Virtual Devices : créer un émulateur
   - Choisissez New
   - Donnez un nom à l'émulateur, Target
   - Cliquez sur Create AVD



Le bundle ADT : tout y est

- Un bundle contenant Eclipse, le plugin ADT est fourni par google
- http://developer.android.com/sdk/installing/bundle.html
- Télécharger et extraire le bundle disponible sous le format d'une archive zip
- Ouvrez Eclipse situé sous : *adt-bundle-<os_platform>/eclipse/*

Android : architecture

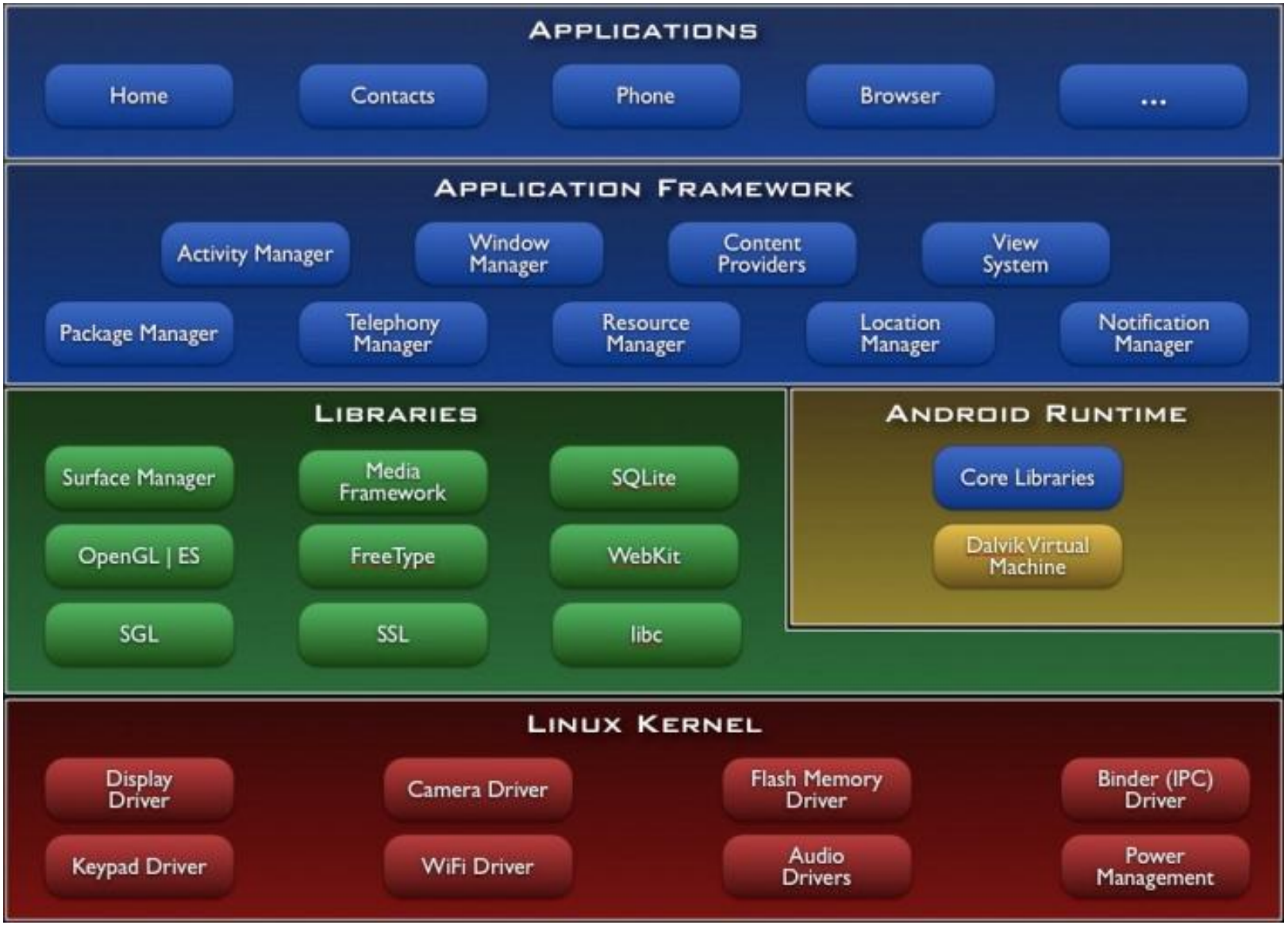
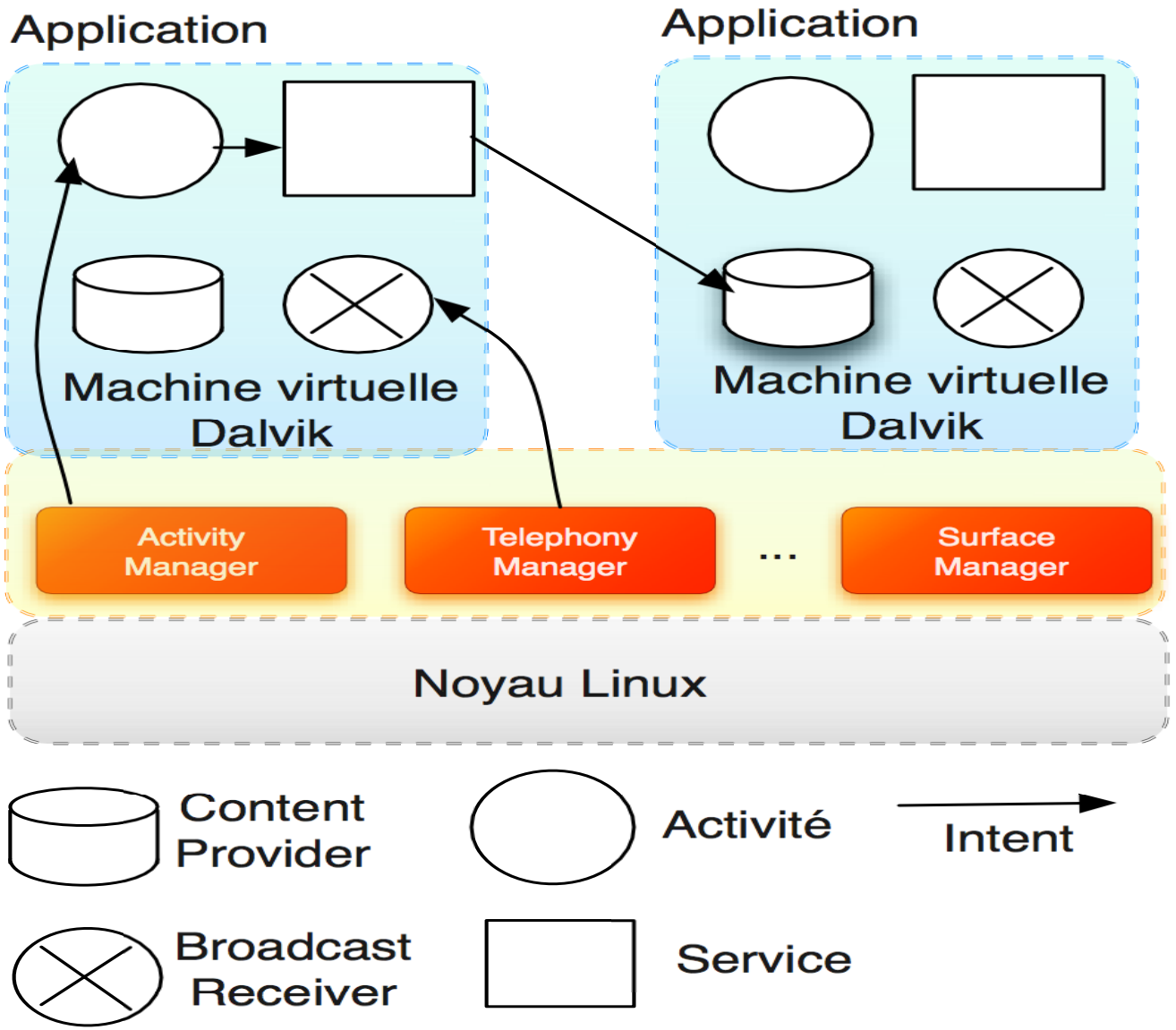


Schéma simplifié



Application Framework

- API publique des fonctions : accessible pour développer des applications tiers
  - Content Providers : accèder aux données d'une applictation (répertoire téléphonique par exemple)
  - Resource Manager : accèder aux ressources de type graphiques, aux fichiers de gabrit (layout) et aux chaines de caractères
  - Notification Manager : affichage des messages dans la barre d'état
  - Activity Manager : gérer le cycle de vie des applications

Bibliothèques

- Ensemble de bibliothèques utilisées par les différents composants d'Android
- Exposées via l'Application Framework
  - Bibliothèque C : libc
  - Bibliothèques multimédias : jouer, enregistrer différents format audio, vidéo MPEG4, H.264, MP3, AAC, AMR, JPG
  - Surface Manager : gérer l'accès au système d'affichage
  - LibWebCore : moteur d'un navigateur web
  - SQLite : un moteur de base de données relationnelles disponible aux applications

Android Runtime

- Machine virtuelle Dalvik
  - Version optimisée de la machine virtuelle Java
  - Support des équipements à faible mémoire
  - Format des classes .dex
  - Utilise le noyau Linux pour la gestion des threads et la mémoire
- Bibliothèques Java
- Chaque application s'exécute dans son propre processus et sa propre instance d'une machine virtuelle

Construction d’une application

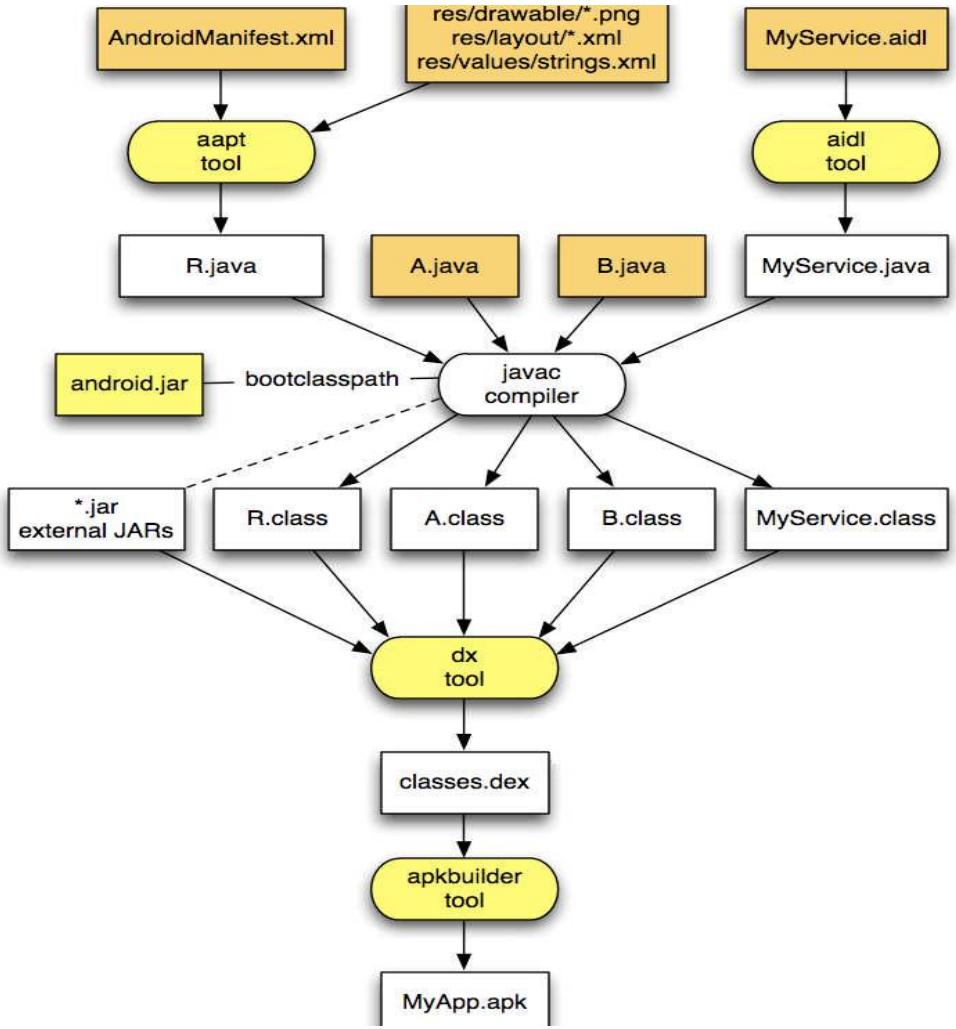
Construction d'une application

Code source

Byte Code Java

Byte code optimisé
Génération fichier.dex

Construction du package
destiné à être installé



Applications Android

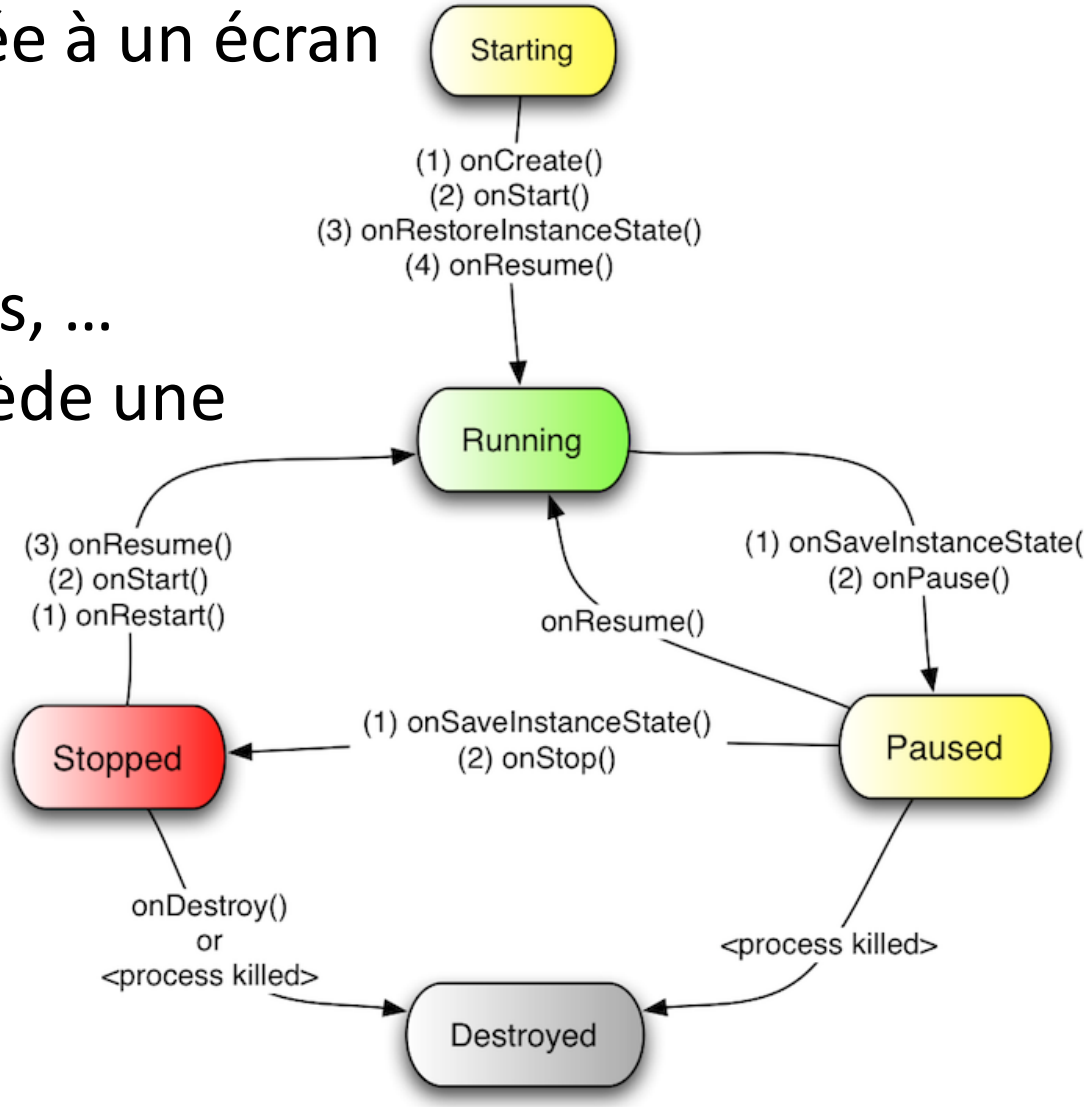
- Application Android : fichier archive (apk)
  - Fichiers classes .dex
  - Ressources : graphiques, gabarits, fichiers de chaînes de caractères
  - Fichier AndroidManifest.xml : description des activités, permissions, services
- Différents composants
  - Activités, Intents, Services, Broadcast Receivers, Content Providers

Les composants

- **Une activité** représente un écran unique avec une interface utilisateur. Par exemple une application gérant les mails aura une activity montrant la liste des emails, une autre pour lire les mails et une autre pour en créer de nouveaux.
- **Services** est un composant qui s'exécute en tâche de fond pour réaliser des opérations longues ou pour effectuer un appel à une tâche distante (appel REST par exemple). Contrairement à une activité un service ne propose pas de d'interface utilisateur.
- **Un content manager** gère les données applicatives. Vous pouvez stocker les données dans n'importe quel système de stockage accessible par votre application (file system, base de données SQLite). A travers le content manager d'autres applications peuvent requêter ou modifier les données (votre application peut par exemple accéder si elle a l'autorisation de l'utilisateur à la liste des contacts.
- **Broadcast Receiver** permet de traiter les différents signaux émis par le système. Par exemple un broadcast annonce que l'écran a été éteint, que la batterie est faible, qu'une image vient d'être prise…. Même si ces composants n'affichent pas d'interfaces, ils peuvent interagir avec la barre de statut pour avertir l'utilisateur qu'un évènement broadcast intervient.

Les activités

- Unité d'exécution associée à un écran d'affichage
  - Plusieurs views
  - Texte, bouttons, tables, …
- Chaque application possède une activité principale
- Différents états
  - Démarage
  - En exécution
  - En pause
  - En arrêt
  - Détruite

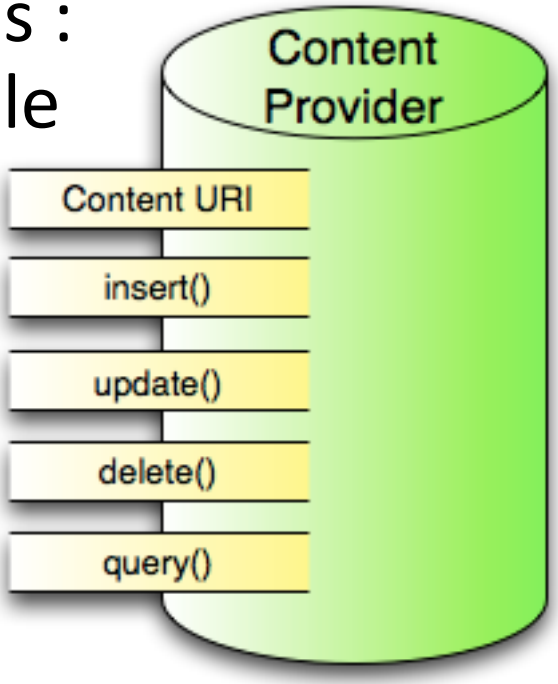


Les services

- Unité d'exécution en arrière plan
- Pas besoin d'avoir une interaction utilisateur : pas d'écran d'affichage
- Exemple : lecteur de musique
  - Affichage de la liste, choix utilisateur, démarrer lecture (activité)
  - L'utilisateur exécute une autre application: l'acitivité est mise en pause
  - Un service continue de jouer de la musique en arrière plan

Les content providers

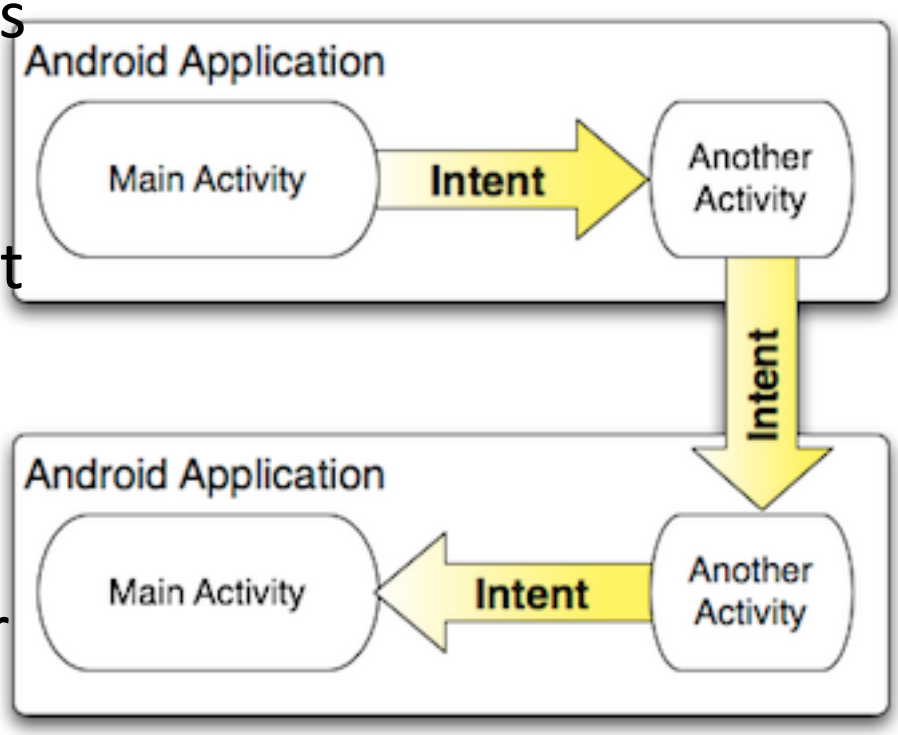
- Les applications stockent des données : base de données SQLite, carte SD ou le téléphone
- Exposer ces donnés aux autres applications : content provider
  - Créer son propre “content provider”
  - Étendre un “content provider” existant



Content Provider	Intended Data
Browser	Browser bookmarks, browser history, etc.
CallLog	Missed calls, call details, etc.
Contacts	Contact details
MediaStore	Media files such as audio, video and images
Settings	Device settings and preferences

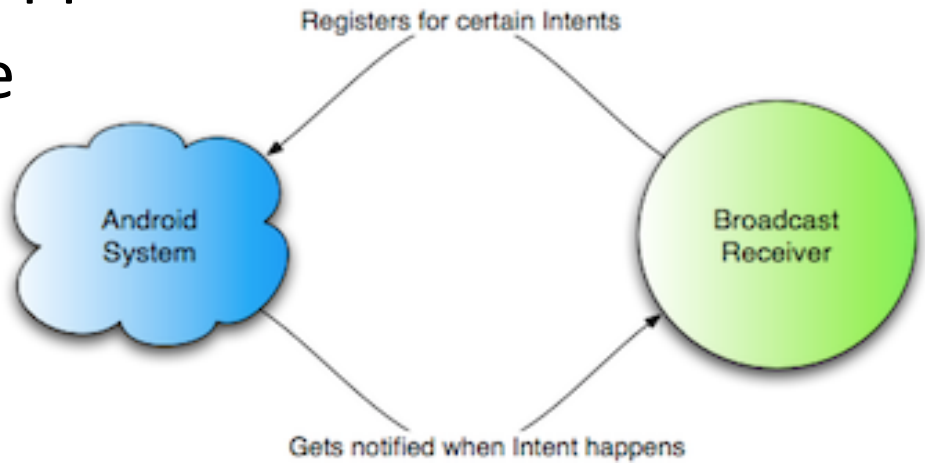
Les intents

- Messages échangés entre les différents blocks applicatifs
- Démarrer une activité, un service ou faire un broadcast
- Asynchrones : pas d'attente
- Explicite : dédié à un composant particulié
- Implicite : type de récépteur



Broadcast Receivers

- Modèle publish/subscribe
- Récepteur : code dormant en attente d'un événement
- Pas d'activité visuelle associée
- Le système diffuse différents événements
  - Arriver d'un SMS, d'un appel
  - Niveau de battérie faibe
  - Démarrage de système



Contexte d’une application

- Activités, services, broadcast receivers : contexte d’une application
- Créer au démarrage d’un composant de l’application
- Informations relatives à l’application Anroid : ressources, classes, Intents, …
- Durée de vie de contexte = durée de vie de l’application
- Context.getApplicationContext ou Activity.getApplication

AndroidManifest.xml

- Chaque application possède un fichier AndroidManifest.xml
  - Ensemble des activités (les intent associés)
  - Ensemble des services (les intent associés)
  - Ensemble des Receivers (les intent associés)
  - Ensemble de permissions pour accèder au matériel et aux composants des autres applications
  - Déclarer la version minimale réquise des API Android

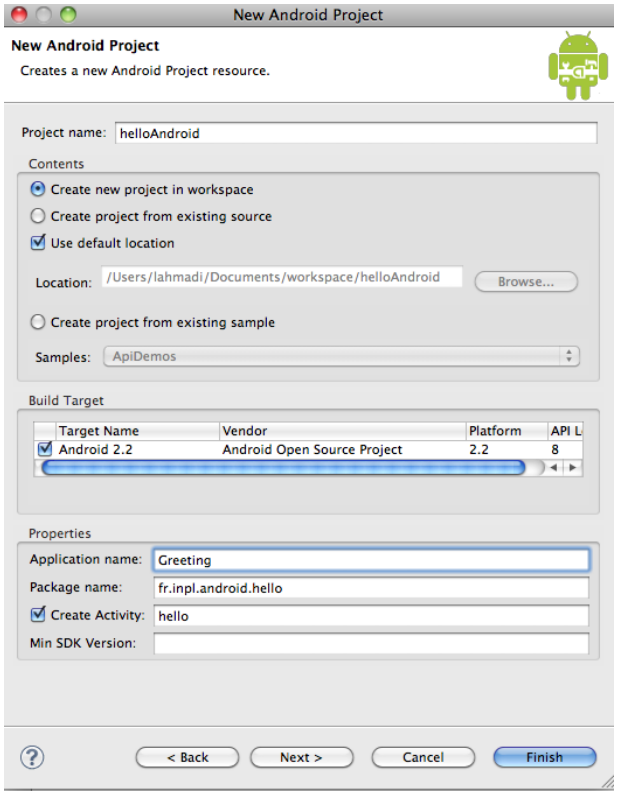
Les permissions

- Modèle de sécurité basé sur les permissions
- Contrôler l'accès aux ressources du système
  - Effectuer des appels, activer le vibreur, accès au carnet d'adresse, accès réseau, GPS, …
- Déclarer puis utiliser : c'est au développeur d'exprimer dans le AndroidManifest.xml les permissions sollicitées

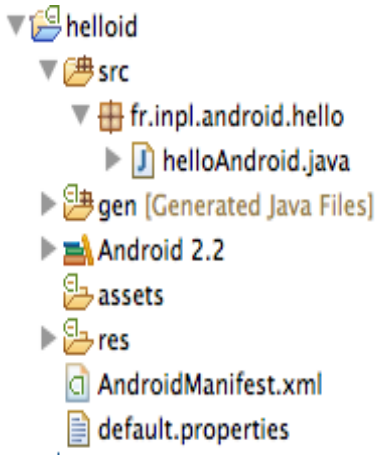
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

Un petit exemple – «Hello, Android»

- Créer un nouveau projet : helloAndroid
  - Cliquer sur File>New>project et choisissez « Android project», puis Next
  - Indiquez les propriétés du projet
    - Nom
    - Target
    - Nom de l'application Android
    - Nom du package Java
    - Créer une première activité



Hello Android




```
package fr.inpl.android.hello;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

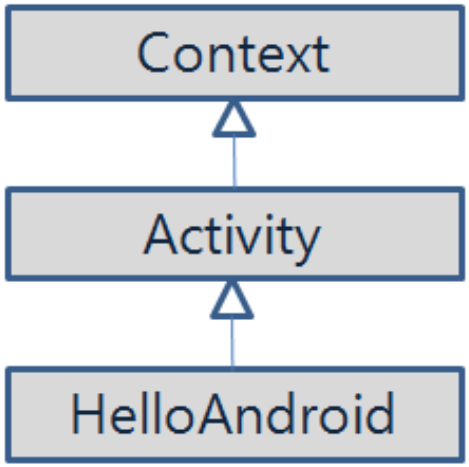
Hello Android : quelques explications

- Un objet TextView

  TextView tv = new TextView(this);

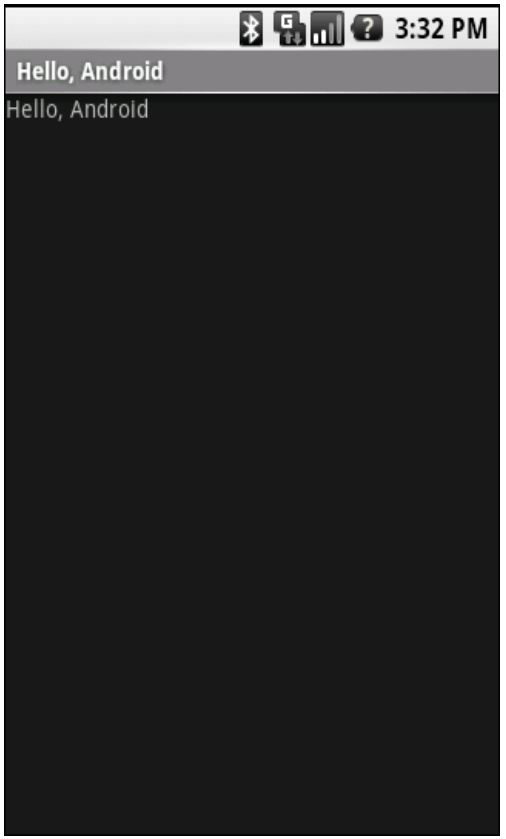
  – Un élément de l'interface utilisateur pour afficher un texte : package Widget

  – Argument : une instance de la classe Context



Hello Android : quelques explications

- Afficher une chaîne de caractères
  tv.setText(‘’ Hello, Android‘’);
- Afficher l’objet tv sur l’écran
  setContentView(tv);
- Test
  - Choisissez Run> Run Configuration
  - Cliquez sur « Android Application »
  - Cliquez sur l’icône
  - Cliquez sur le bouton « Browse»
  - Sélectionnez votre projet Android
  - Ensuite « Apply», « Run »

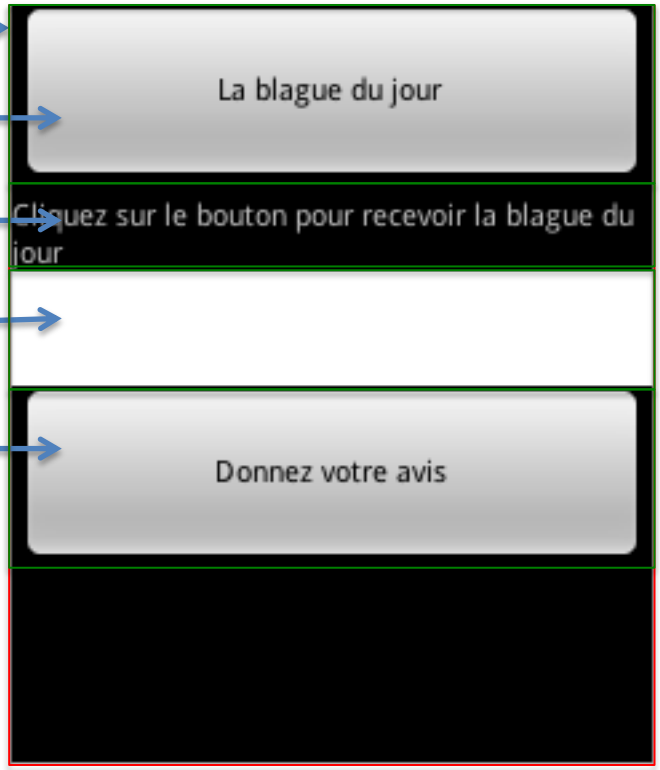


La blague du jour

- Récupérer et afficher une blague du site : http://www.blague.info/blagues-du-jour/blagues.php
- Afficher la blague
  - Un web view
- Noter la blague
  - étoiles d'appréciation

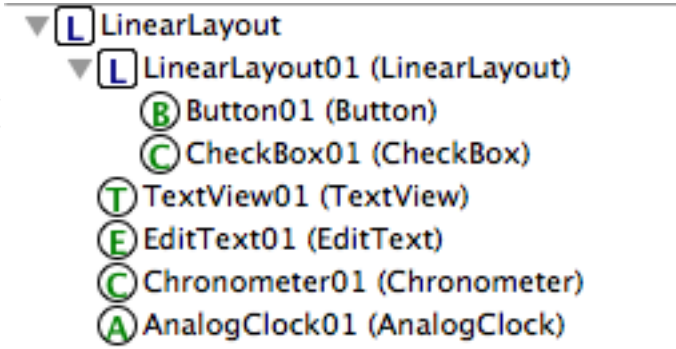
Interface Utilisateur (1)

- Interfaces graphiques : fichiers XML
- Éditeur fourni par le plugin ADT
- Fichier : layout/main.xml
  - LinearLayout
  - Button
  - WebView
  - RatingBar
  - Button



Éléments d'une interface graphique

- Composants graphiques
  - Layouts : Gabarits de positionnement
    - Linéaire : verticale ou horizontale
    - GridView
    - ListView : pour créer des listes
  - Views
    - Button
    - CheckBox
    - EditText
    - TextView
    - Chronometer
    - RatingBar



Élément de UI : propriétés

- Chaque composant graphique possède un ensemble de propriétés
  - LinearLayout : orientation, gravity, height, width
  - Views
    - Id : @+id/identifiant à utiliser dans le code pour lui associer un objet Java
    - Text : @string/identifiant ou texte, valeur à afficher comme texte associé
    - Layout height : wrap_content, fill_parent, match_parent
    - Layout width : wrap_content, fill_parent, match_parent
    - Layout weight : un ratio de taille entre les différents éléments du layout

Le fichier ressource : strings.xml

- Fichier XML sous : res/values/strings.xml
  - Basée sur la locale du système
  - /res/values-fr, /res/values-en : affichage des valeurs dans la langue correspondante

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
</resources>
```

- Identifiant du texte d'affichage d'un TextView

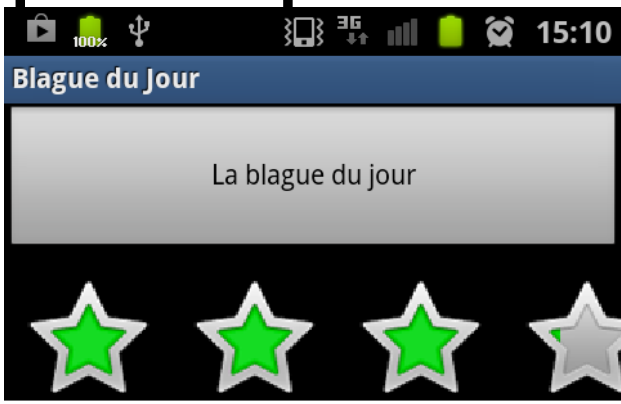
```xml
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

- Dans un code Java

```java
String string = getString(R.string.hello);
```

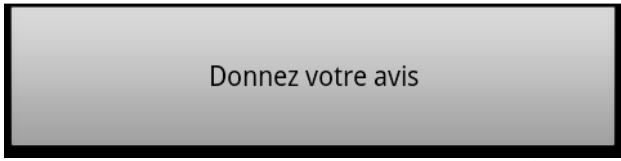
Blague du jour : Activité principale

```java
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.RatingBar;
import android.widget.Toast;

public class principale extends Activity {
    private WebView webBlague;
    private Button btnBlague;
    private Button btnAvis;
    private RatingBar rate;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webBlague = (WebView) findViewById(R.id.webbalgue);
        rate = (RatingBar) findViewById(R.id.RatingBar02);
        btnBlague = (Button) findViewById(R.id.BtnBlague);
        btnAvis = (Button) findViewById(R.id.BtnAvis);
        btnBlague.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                webBlague.loadUrl("http://www.blague.info/blagues/humour/drole-118
            }

        });
        btnAvis.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                float value = rate.getRating();
                Toast toast = Toast.makeText(getApplicationContext(), "Votre avis est : "+value, Toast.LENGTH_SHORT);
                toast.show();
            }

        });
    }
}
```



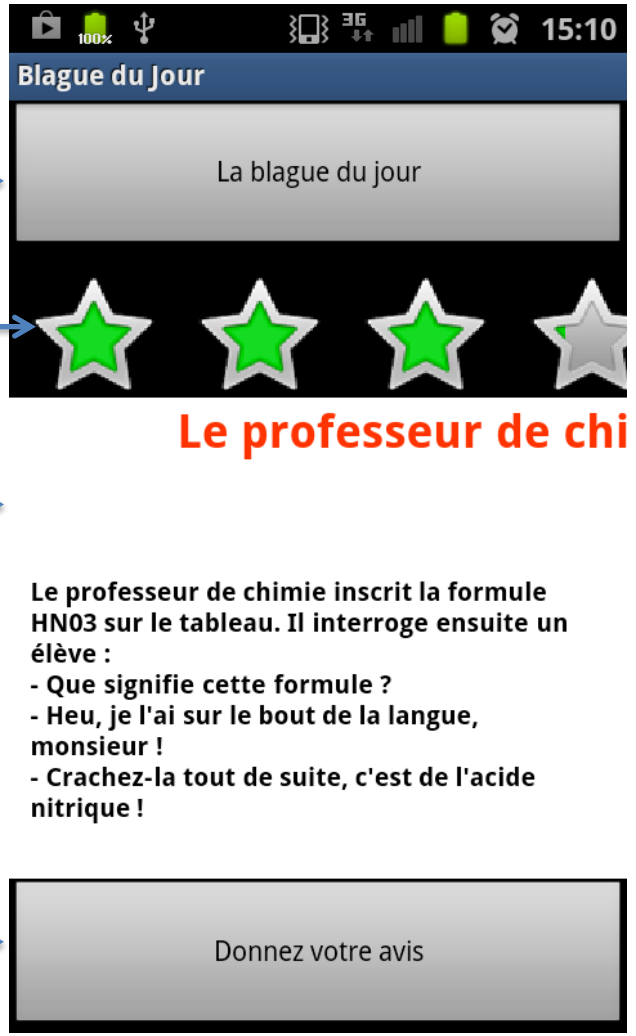
Le professeur de chi

Le professeur de chimie inscrit la formule HN03 sur le tableau. Il interroge ensuite un élève :
- Que signifie cette formule ?
- Heu, je l'ai sur le bout de la langue, monsieur !
- Crachez-la tout de suite, c'est de l'acide nitrique !



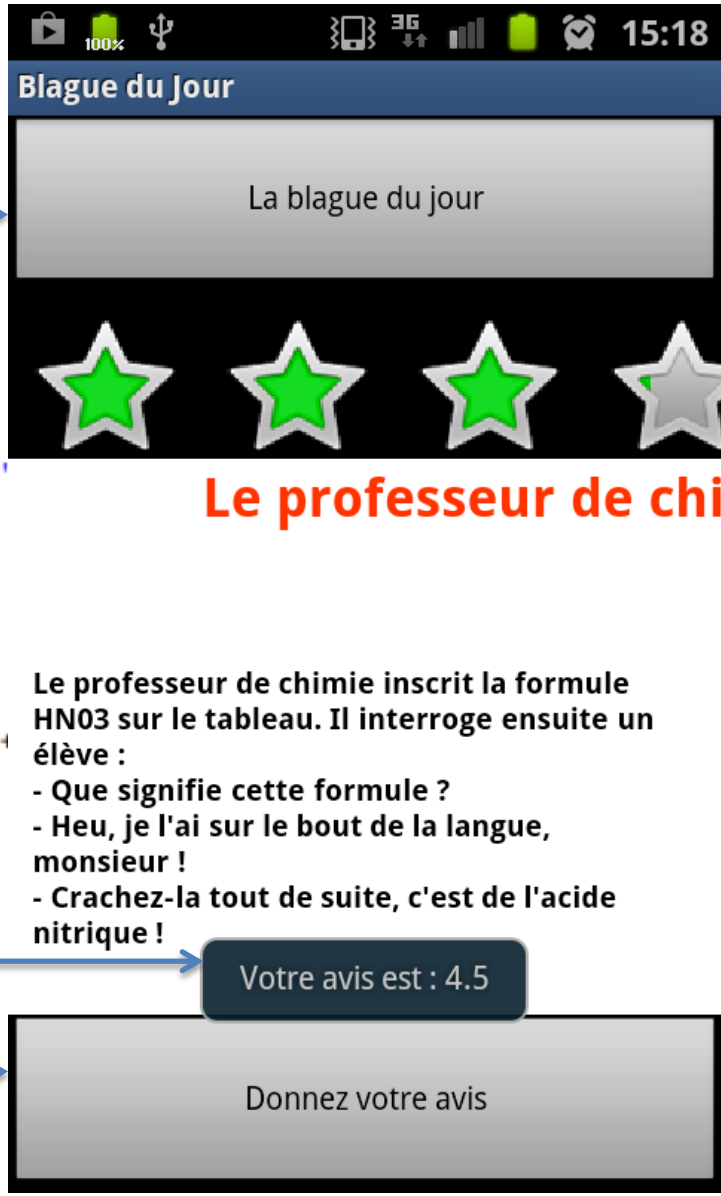
Blague du jour : les objets

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.RatingBar;
import android.widget.Toast;

public class principale extends Activity {
    private WebView webBlague;
    private Button btnBlague;
    private Button btnAvis;
    private RatingBar rate;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        webBlague = (WebView) findViewById(R.id.webbalgue);
        rate = (RatingBar) findViewById(R.id.RatingBar02);
        btnBlague = (Button) findViewById(R.id.BtnBlague);
        btnAvis = (Button) findViewById(R.id.BtnAvis);
```



Blague du jour : les événements

```
btnBlague.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        webBlague.loadUrl("http://www.blague.info/blagues/humour/drole-11878.html"
    }

});
btnAvis.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        float value = rate.getRating();
        Toast toast = Toast.makeText(getApplicationContext(), "Votre avis est : "
        toast.show();
    }

});
```



AndroidManifest.xml

- Renseignements sur l'application Android
  - Nom de l'application : *@string/app_name*
  - Nom du package : *fr.inpl.android.blague*
  - Composants : activity, services, receivers, provides, intent (action, category), etc
  - Permissions sollicitées par l'application

```
webBlague.loadUrl("http://www.blague.info/blagues/humour/drole-11878.html");
```

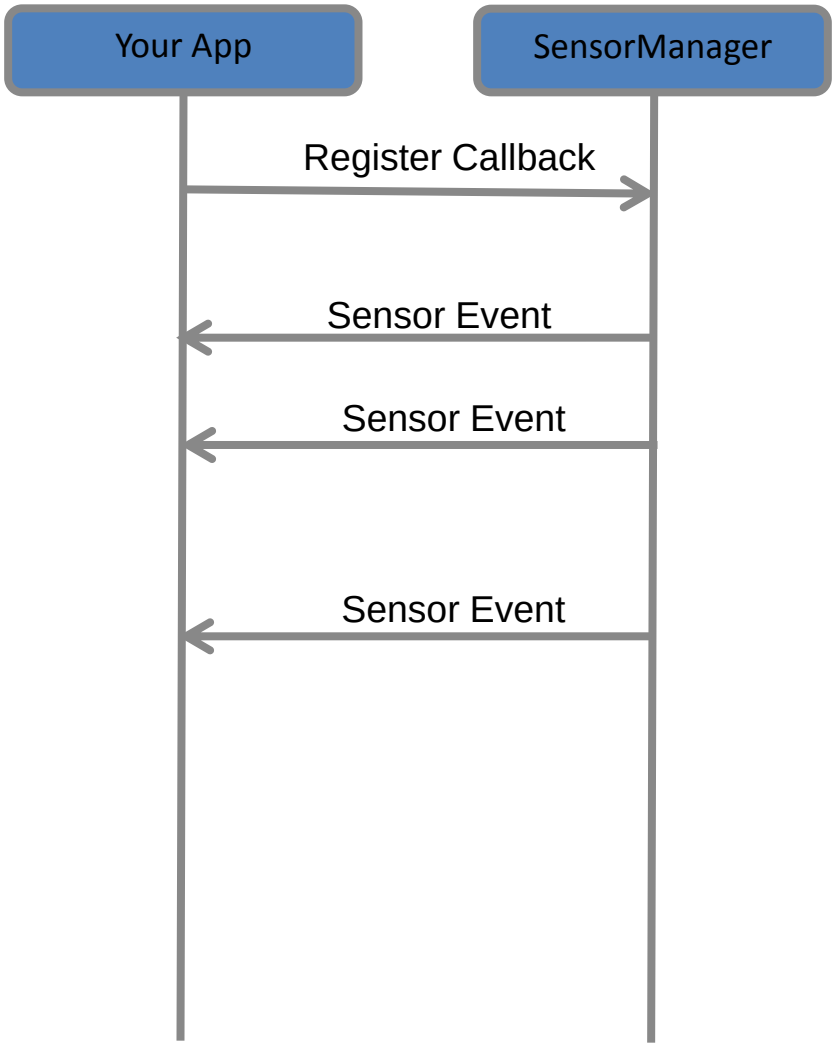
<uses-permission android:name="android.permission.INTERNET"></uses-permission>

Les senseurs dans Android

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a[1]	n/a[1]
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes[2]	Yes[2]	Yes[2]	Yes
TYPE_PRESSURE	Yes	Yes	n/a[1]	n/a[1]
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes[2]	Yes	Yes	Yes

Appels asynchrones

- Les sensors sont contrôlées par des services externes : ils envoient seulement des évènements
- L'application doit enregistrer des callbacks pour recevoir les évènements d'un sensor
- Chaque sensor possède une interface XXXListener à implémenter par la classe contenant les callbacks : LocationListener



Faire appel au Service Système

- Les sensors sont gérés par des les classes XXXManager :
  - LocationManager (GPS)
  - SensorManager (accéléromère, Gyro, …)
- Obtenir une référence de la classe XXXManager en utilisant la méthode getSystemService()

```
public class MyActivity … {

  private SensorManager sensorManager_;

  public void onCreate(){
     …

     sensorManager_ = (SensorManager) getSystemService(SENSOR_SERVICE);
  }

}
```

Recevoir les mises-à-jour de la localisation

- Le LocationManager gère les souscriptions aux mises-à-jour de la localisation via le GPS ou le réseau
- Nécessite l'implémentation de l'interface *LocationListener*
- S'enregistrer en utilisant la méthode *requestLocationUpdates*
  - Précision de la localisation (mètres)
  - Fréquence des mises – à jour (millisecondes)

```
public class MyActivity … implements LocationListener{

  private LocationManager locationManager_;

  public void onCreate(){
     …

     locationManager_ = (LocationManager) getSystemService(LOCATION_SERVICE);
     locationManager_.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10,
                                    Criteria.ACCURACY_FINE, this);
  }

}
```

Fournisseurs de la localisation

- Plusieurs fournisseurs sont disponibles
  - GPS
  - Réseau
- Le GPS consomme plus de batterie mais il est plus précis
  - GPS : 25 secondes * 140mA = 1mAh
  - Réseau : 2 secondes * 180mA = 0.1 mAh
- Vous pouvez utiliser le PASSIVE_PROVIDER pour mettre à jour la localisation

```
public class MyActivity … implements LocationListener{

  private LocationManager locationManager_;

  public void onCreate(){
    …

    locationManager_ = (LocationManager) getSystemService(LOCATION_SERVICE);
    locationManager_.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 10,
                                    Criteria.ACCURACY_FINE, this);
  }

}
```

L’interface LocationListener

```
public class MyActivity … implements LocationListener{

  …
        // Called when your GPS location changes
        @Override
        public void onLocationChanged(Location location) {

        }

        // Called when a provider gets turned off by the user in the settings
        @Override
        public void onProviderDisabled(String provider) {


        }

        // Called when a provider is turned on by the user in the settings
        @Override
        public void onProviderEnabled(String provider) {


        }

        // Signals a state change in the GPS (e.g. you head through a tunnel and
        // it loses its fix on your position)
        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {


        }
}
```

Information de Localisation

```
public class MyActivity … implements LocationListener{

  …
        // Called when your GPS location changes
        @Override
        public void onLocationChanged(Location location) {

                double altitude = location.getAltitude();
                double longitude = location.getLongitude();
                double latitude = location.getLatitude();
                float speed = location.getSpeed();
                float bearing = location.getBearing();
                float accuracy = location.getAccuracy(); //in meters
                long time = location.getTime(); //when the fix was obtained

                // Other useful Location functions:
                //
                // location.distanceTo(dest)
                // location.bearingTo(dest)
        }


}
```

Désinscription

- Il faut désinscrire les mises à jour lorsque l'activité est en pause : valable pour tous les sensors

```
public class MyActivity … {

        private LocationManager locationManager_;

        public void onCreate(Bundle savedInstanceState) {
                …
                locationManager_ = (LocationManager)getSystemService(LOCATION_SERVICE);
        }
        protected void onPause() {
                super.onPause();
                locationManager_.removeUpdates(this);
        }
        protected void onResume() {
                super.onResume();
                locationManager_.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10,
                                                        Criteria.ACCURACY_FINE, this);
        }
…
}
```

Les autres sensors

```
public class MyActivity … implements SensorListener{
  private Sensor accelerometer_;
  private SensorManager sensorManager_;

  public void connectToAccelerometer() {
        sensorManager_ = (SensorManager)getSystemService(SENSOR_MANAGER);
        accelerometer_ = sensorManager_
                              .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorManager_.registerListener(this, accelerometer_,
                              SensorManager.SENSOR_DELAY_NORMAL);


  }
   // Called when a registered sensor changes value
        @Override
        public void onSensorChanged(SensorEvent sensorEvent) {
              if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
                    float xaccel = sensorEvent.values[0];
                    float yaccel = sensorEvent.values[1];
                    float zaccel = sensorEvent.values[2];


              }
        }
        // Called when a registered sensor's accuracy changes
        @Override
        public void onAccuracyChanged(Sensor arg0, int arg1) {
              // TODO Auto-generated method stub

        }
```

Autre approches : plusieurs capteurs

```
public class MyActivity … {

        private class AccelListener implements SensorListener {
                        public void onSensorChanged(SensorEvent sensorEvent) {
                                …
                        }
                        public void onAccuracyChanged(Sensor arg0, int arg1) {}
                }

         private class LightListener implements SensorListener {
                        public void onSensorChanged(SensorEvent sensorEvent) {
                                …
                        }
                        public void onAccuracyChanged(Sensor arg0, int arg1) {}
                }

                private SensorListener accelListener_ = new AccelListener();
                private SensorListener lightListener_ = new LightListener();

                …
                public void onResume(){
                   …
                 sensorManager_.registerListener(accelListener, accelerometer,
                                        SensorManager.SENSOR_DELAY_GAME);
                 sensorManager_.registerListener(lightListener, lightsensor,
                                        SensorManager.SENSOR_DELAY_NORMAL);

                }
                public void onPause(){
                  sensorManager_.unregisterListener(accelListener_);
                   sensorManager_.unregisterListener(lightListener_);
                }
```

Mettre à jour le GUI

- Utilisation des threads et des handlers pour mettre à jour l'interface graphique avec les valeurs obtenues depuis les capteurs

```
public class MyActivity … implements SensorListener{
        private class AccelWork implements Runnable {
            private Location data_;
             public AccelWork(Location d){data_ = d;}

             public void run(){
                  //do something with the data to the GUI
            }
        }

         private Handler myHandler_ = new Handler();

        // Called when a registered sensor changes value
         @Override
         public void onSensorChanged(SensorEvent sensorEvent) {
             AccelWork work = new AccelWork(sensorEvent);
            myHandler_.post(work);
         }
}
```

Conclusion

- Développement Android
  - Programmation JAVA, plugin complet et gratuit
  - Nécessite de bonnes connaissances en développement Java

Pour aller plus loin …

- NDK : Native Development Kit
  - Inclusion des bibliothèques et d'applications natives
  - Portions du code en C/C++
  - Interfaces JNI pour les appels depuis le code Java
  - Améliorer les performances de certaines fonctionnalités

http://developer.android.com/sdk/ndk/index.html